Technische Fakultät
Albert-Ludwigs-Universität, Freiburg
Lehrstuhl für Kommunikationsysteme
Prof. Dr. Gerhard Schneider

Bachelorarbeit

Digital Forensics of Old Floppy Disk Bit Streams

9. Juni 2012

betreut durch
Dr. Dirk von Suchodoletz

Erstprüfer
Prof. Dr. Gerhard Schneider

Richard Schneider
Matr.-Nr.: 2711155

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Ort, Datum Unterschrift

Inhaltsverzeichnis

1. Einleitung **1**

2. Theoretische Grundlagen **3**
2.1. Diskette 3
2.2. Low-Level Formatierung 4
2.3. High-Level Formatierung 5
2.4. Dateisystem 5
2.5. Speicherorganisation 6
2.6. Löschen von Dateien 6

3. BTOS/CTOS Dateisystem **7**
3.1. Volume Home Block 8
3.2. Master File Directory 8
3.3. File Header Block 8
3.4. Bad Block 9
3.5. Allocation Bit Map 9

4. Ansatz **11**

5. Durchführung **13**

6. Ergebnisse **15**

7. Zusammenfassung und Ausblick **19**

Anhang **21**
A. DVD 21

Literaturverzeichnis **23**

1. Einleitung

Zur Erhaltung der Verfügbarkeit von gespeicherten Informationen, werden in dieser Arbeit mit den Methoden der digitalen Forensik Dateien von alten Disketten gesichert.

Während Informationen auf Tontafeln tausende Jahre überdauern konnten und auch heute noch entzifferbar sind, treten Schwierigkeiten mit der Lesbarkeit digital gespeicherter Informationen schon bereits nach wenigen Jahrzehnten auf. Die stark begrenzte Haltbarkeit der Datenträger, veraltete Dateiformate und veraltete Systeme führen nach und nach zu erschwertem Datenzugang bis hin zum kompletten Datenverlust Sietmann [2006]. Diese Problematik wird „digitales Vergessen“ genannt. Diese Arbeit befasst sich praktisch mit dem Problem des „digitalen Vergessens“. Das Ziel ist es, so viele Daten wie nur möglich zu extrahieren und den Zugriff auf einem aktuellen System zu erhalten. Für die Durchführung dieses Experiments sind das Wissen und die Verfahrensweisen der digitalen Forensik wichtige Hilfsmittel.

Die digitale Forensik dient der Aufklärung von Straftaten, deren Ausübung in Verbindung mit digitalen Geräten stehen. Die Aufgaben sind die Beweismittelsicherung von digitalen Systemen, eine Hypothesenbildung zum Tathergang auf Basis der gesicherten Indizien und die Überprüfung dieser Hypothesen durch weitere digitale Beweismittel. Beweismittel sind in diesem Zusammenhang digitale Objekte, welche verlässliche Informationen enthalten und dabei helfen, Hyphothesen zu widerlegen oder zu untermauern. An einen realen Tatort angelehnt beinhaltet die digitale Ermittlung die Phasen der Systemsicherung, Beweismittelsuche und Ereignisrekonstruktion Carrier [2005].

Da es sich bei der zugrundeliegenden Problematik um die Wiederherstellung von Daten und nicht um die Aufklärung einer Straftat handelt, wird ausschließlich auf die hierfür interessanten Phasen der Systemsicherung und der Beweismittelsuche eingegangen.

Während der Systemsicherung wird der Zustand des Systems konserviert. Das Ziel ist es, so wenige Daten wie möglich zu verändern und den Zustand für weitere Untersuchungen zu erhalten Carrier [2005]. Dieser Schritt wurde bereits mit Kryoflux Bartsch [2011] durchgeführt. Kryoflux ist ein Diskettenlaufwerkcontroller mit einer USB-Schnittstelle, der speziell für das präzise Auslesen von Disketten entwickelt wurde. Während des Auslesevorgangs wird der magnetischen Fluss der Diskette gelesen und dieses Signal in eine Abbilddatei auf dem Hostcomputer geschrieben. Die dazugehörige Software macht es möglich, die Abbilder in Standarddiskettenformate zu exportieren.

Als Teil der Beweismittelsuche steht die forensische Untersuchung hier im Vordergrund. Sie dient der Sammlung von Informationen und ist auf verschiedenen Abstraktionsebenen möglich Carrier [2005]. Die erhaltenen Informationen werden anschließend zur Wiederherstellung der Daten genutzt. Die Beschriftungen auf den Hüllen geben erste Eigenschaften der Dateinträger preis. Es handelt sich um Quad Density, Double Sided, 5.25 Zoll Disketten mit einer Spurdichte von 96 Spuren pro Zoll. Durch vorausgehende Analysen auf der Hardwareebene ist bekannt, dass die Disketten in dem damals weit verbreiteten MFM Format formatiert und in 16 Sektoren mit 256 Bytes pro Sektor unterteilt sind. Darauffolgende Untersuchungen der Diskettenabbilder unter Verwendung eines Hexeditors lassen die wiederkehrenden Zeichenketten „sysImage.sys“, „badBlk.sys“ und „log.sys“ erkennen. Diese Zeichenketten geben Hinweise auf das verwendete BTOS/CTOS Dateisystem, welches von Computern mit dem Burroughs Technologies Operating System oder dessen Nachfolger Convergent Technologies Operating System benutzt wird. Weiterhin ist bekannt, dass abgesehen von den ursprünglichen Systemen derzeit kein Betriebssystem oder Programm existiert, welches die Disketten oder deren Abbilder mit diesem Dateisystem richtig interpretieren kann. Es existiert auch kein Emulator für diese Systeme. Lediglich die Beschreibung des Dateisystems ist vorhanden.

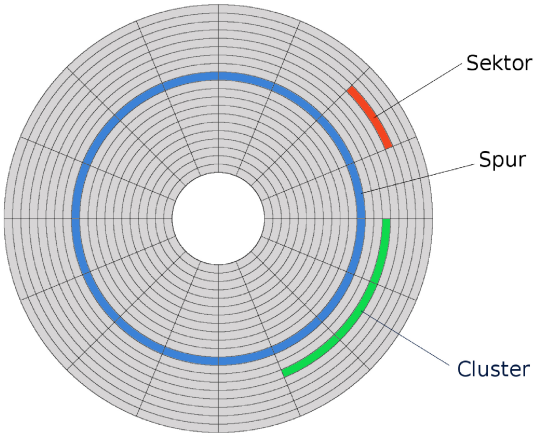
Als Grundlage dienen Disketten des neuseeländischen Maori Land Courts und der US Küstenwache. Die weiteren Analysen der Diskettenabbilder auf der Dateisystemebene und die Extraktion der Daten sind die fehlenden Schritte zur Wiederherstellung der Dateien und somit Grundlage dieser Arbeit.

2. Theoretische Grundlagen

Für die folgende Untersuchung der Abbilder sind grundlegende Kenntnisse über den Aufbau von Disketten und Dateissystemen notwendig. Dazu werden zunächst Charakteristika von Disketten erläutert. Daraufhin stehen allgemeine Konzepte von Dateisystemen bis hin zu Wissen um Vorgänge beim Speicher- und Löschvorgang im Fokus.

2.1. Diskette

(a) 5.25 Zoll Diskette mit 360 KiB Speicherkapazität Frank [2005].



(b) Logischer Aufbau der internen Speicherscheibe.

Abbildung 2.1.

Eine Diskette speichert Informationen auf einer mit magnetisierbarem Material beschichteten, flexiblen Kunststoffscheibe. Dieses Speichermedium wird zum Schutz von einer quadratischen Kunststoffhülle umgeben, wie in Abbildung **2.1a** erkennbar ist. Da während der Rotationen im Betrieb der Schreib/Lese Kopf des Laufwerks im

Gegensatz zu Festplatten direkt auf der Scheibe aufliegt, und diese auch Kontakt zur Schutzhülle hat, kommt es zu einer Verringerung der Lebensdauer, die zwischen 5 und 30 Jahren liegt. Es kommen 8 Zoll bis 3,5 Zoll Formate mit 80 KiB bis zu 200 MiB Speicherkapazität vor. Die physikalischen Eigenschaften einer Diskette werden beschrieben durch die vom beschichteten Material abhängige Aufzeichnungsdichte und der Anzahl der beschreibbaren Seiten. Diese Eigenschaften wirken sich auf die mögliche Spurdichte, die Zahl der Spuren, die Anzahl der Sektoren und deren Größe aus. Die verwendete Codierung gibt an, wie Informationen auf dem Speichermedium repräsentiert werden. Diese Faktoren beeinflussen die Speicherkapazität. Sektoren sind die kleinsten adressierbaren Datenblöcke und können die Größen von 128 Byte bis 1024 Byte besitzen. Aufgrund der Tatsache, dass innere Spuren kürzer sind als äußere, ist es möglich, auf den äußeren eine höhere Anzahl an Sektoren unterzubringen. Technisch gesehen war es jedoch einfacher, die gleiche Anzahl an Sektoren pro Spur zu verwenden, wodurch Schwierigkeiten, verursacht durch die verschiedenen relativen Geschwindigkeiten des Schreib/Lese Kopfes, vermieden werden konnten Tanenbaum und Goodman [1987].

2.2. Low-Level Formatierung

Damit auf einer Diskette Informationen gespeichert werden können, muss eine einheitliche, physikalische Aufteilung in Spuren und Sektoren, wie in Abbildung **2.1b** gezeigt, vorgenommen werden. Dieser Vorgang wird Low-Level Formatierung genannt. Die erste Aufteilung erfolgt durch den Hersteller und kann weiterhin auch vom Laufwerkcontroller ausgeführt werden. Sektoren enthalten einen Abschnitt für Header, Daten und Fehlerkorrekturcode. Die Codierung, in der die Daten physikalisch auf der Diskette gespeichert sind, ist auch vom Controller abhängig. Eine in dieser Zeit für Festplatten und Disketten weit verbreitete Codierung ist MFM Tanenbaum [2001].

Der nächste logische Punkt ist die Partitionierung. Dieser Teil wird ausgelassen, da Disketten immer als eine Partition angesehen werden, und daher dieser Teil für die weitere Betrachtung nicht wichtig ist Tanenbaum [2001].

2.3. High-Level Formatierung

Die High-Level Formatierung ist eine logische Formatierung, die auf der Low-Level Formatierung aufsetzt und den Datenträger für das Betriebssystem nutzbar macht. Dafür wird auf dem Datenträger ein Dateisystem eingerichtet. Das Dateisystem stellt eine Verwaltungsstruktur dar. Es ermöglicht die Langzeitspeicherung und das Abrufen von Dateien eines Datenträgers. Hierzu nutzt das Dateisystem die Sektoren der Low-Level- Formatierung und fasst diese meist zu größeren Einheiten den sogenannten Clustern zusammen. Ein Cluster ist die kleinste adressierbare Einheit aus der Sicht des Dateisystems Tanenbaum [2001].

2.4. Dateisystem

Das Dateisystem ist ein Teil des Betriebssystems und ermöglicht die Verwaltung und Organisation von Dateien auf einem Datenträger. Es legt die Struktur, Benennung, Zugriff, Benutzung, Schutz und Implementierung von Dateien fest. Betriebssysteme unterstützen meist mehrere Arten von Dateien. Zum Einen gibt es die regulären Dateien, welche in ASCII- und Binärdateien aufgeteilt werden. ASCII-Dateien beinhalten ASCII-Textzeilen, deren Ende durch spezielle Steuerzeichen gekennzeichnet sein kann. Binärdateien sind alle anderen Dateien und enthalten auch nicht lesbaren Text. Zum Anderen existieren Systemdateien wie beispielsweise Verzeichnisse, die der Aufrechterhaltung der Dateisystemstruktur dienen. Zusammengehörige Dateien können für eine bessere Nutzung in Verzeichnisse gruppiert werden. Es gibt verschiedene Arten von Verzeichnisstrukturen. Das Single-Level Directory Konzept stellt nur ein Verzeichnis zur Verfügung. Alle Dateien befinden sich somit im gleichen Verzeichnis. Da alle Dateien eindeutig benannt werden müssen, kann es bei mehreren Benutzern oder vielen Dateien zu Konflikten kommen. Der Two-Level Directory Ansatz ermöglicht die Erstellung von mehreren Verzeichnissen auf der selben Ebene. Jeder Benutzer kann so seine eigenen Dateien in einem separaten Ordner speichern. Die Verschachtelung von Verzeichnissen ist aber nicht möglich. Bei einer großen Menge an Dateien ist es jedoch günstig, hierarchische Verzeichnisstrukturen zu verwenden. Dabei kann ein Verzeichnis mehrere Unterverzeichnisse besitzen und ermöglicht dadurch eine umfassende Organisation der Dateien. Zur Identifikation besitzen Dateien einen Namen und je nach Betriebssystem eine optionale oder erforderliche Erweiterung. Diese Erweiterung kann einen Hinweise auf den vorliegenden Dateityp geben. Für das Betriebssystem ist eine Datei zumeist nur eine Bytesequenz. Neben dem Namen und den Daten der Dateien verknügt das Betriebssystem zusätzliche Informationen mit Dateien. Bei diesen Metainformationen kann es sich beispielsweise um Dateigröße, Erstellungszeit oder den Besitzer der Datei handeln Tanenbaum [2001].

2.5. Speicherorganisation

In welcher Reihenfolge Daten gespeichert werden, hängt von der verarbeitenden CPU ab. Es werden Big Endian, Little Endian und Mischvarianten unterschieden. Zur Anwendung dieser Speicherformate kommt es immer dann, wenn Daten eines von der CPU unterstützten Datentyps gespeichert werden müssen, welche die Länge der kleinsten adressierbaren Einheit der CPU übersteigen. Die am weitesten verbreitete kleinste adressierbare Einheit ist 1 Byte. Im Big Endian Format wird das höchstwertige Byte als erstes gespeichert und bei Little Endian das niederwertigste Byte Carrier [2005].

2.6. Löschen von Dateien

Das Löschen einer Datei umfasst in den meisten Fällen nur die Freigabe der zur Datei gehörigen Speichereinheiten. Dadurch wird dieser Speicherplatz für die Speicherung von anderen Dateien zur Verfügung gestellt. Werden nach dem Löschen die alten Speichereinheiten nicht oder nur teilweise von anderen Dateien überschrieben, besteht die Möglichkeit, Fragmente oder die gesamte gelöschte Datei wiederherzustellen Carrier [2005].

3. BTOS/CTOS Dateisystem

Nach den allgemeinen Grundlagen wird im folgenden Kapitel der spezifische Aufbau des BTOS/CTOS Dateisystems erklärt.

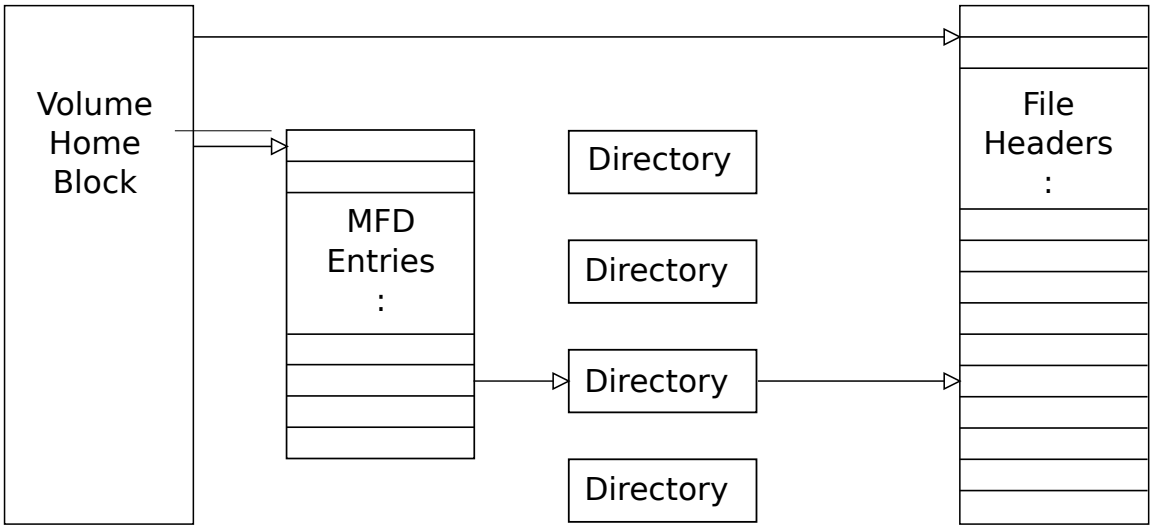


Abbildung 3.1.: Aufbau des BTOS/CTOS Dateisystems.
Convergent Technologies [1987]

Das Basiselement des BTOS/CTOS Dateisystems ist ein 512 Byte Sektor der durch eine 32 Bit lange logische Dateiadresse oder durch die Cylinder/Head/Sector Adressierung lokalisiert wird. Bit 0 bis 29 der Adresse dienen zur Adressierung. Ein oder mehrere zusammenhändende Sektoren werden zusammengefasst in einen Disk Extent, welcher durch seine Anfangsadresse und Länge spezifiziert wird. Dateien werden in Verzeichnisse derart gruppiert, dass eine Datei zu genau einem Verzeichnis gehört. Ein Datenträger enthält jedenfalls ein „sys“ Verzeichnis, worin zumindest die Systemdateien zur Beschreibung des Datenträgers enthalten sind. Die maximale Anzahl an Verzeichnissen und Dateien wird bei der Erstellung des Datenträgers oder der Verzeichnisse festgelegt Convergent Technologies [1987].

3.1. Volume Home Block

Die Hauptstruktur bildet der 256 Byte große Volume Home Block (VHB). Der VHB beinhaltet die Positionen und Größen der weiteren Strukturen wie Master File Directory und File Header Block. Der VHB enthält weiterhin Zeiger auf Systemdateien, wie beispielsweise das Abbild des Betriebssystems, die Systemlogdatei und das Absturzspeicherabbild. Während der Erstellung des BTOS/CTOS Dateisystems wird am Anfang ein aktiver und in der Mitte des Datenträgers ein Backup des VHBs angelegt. Jeder VHB besitzt neben Name, Passwort auch eine Prüfsumme als Hinweis auf dessen Konsistenz Convergent Technologies [1987].

3.2. Master File Directory

Das Master File Directory (MFD) ist aus Speicherabschnitten aufgebaut, die jeweils bis zu 14 Einträge für Verzeichnisse bereitstellen. Diese Einträge beinhalten unter anderem den Namen, das Passwort, die Größe und einen Zeiger auf das Verzeichnis. Der Zeiger führt zum Anfang eines Bereichs, in den die Dateien durch den Hashwert des Dateinamens abgebildet werden. Es kann nur durch das MFD auf Verzeichnisse zugegriffen werden, da sie keine eigenen Einträge in den File Headern haben Convergent Technologies [1987].

3.3. File Header Block

Jeder File Header beschreibt eine Datei. Im File Header Block (FHB) werden primäre und optional sekundäre File Header gespeichert. Die sekundären File Header schließen in einem bestimmten, im VHB definierten, Abstand an die primären File Header an. Die sekundären File Header dienen als Sicherungskopie. Zu den Metainformationen der Datei gehören Dateiname, Dateigröße, Passwort, Erstellungs-, Zugriffs- und Veränderungszeit. Falls die Länge des Dateinamens 0 beträgt, ist ein File Header inaktiv. Die eigentlichen Daten der Datei sind aufgeteilt in Disk Extents. Die Zeiger auf die Disk Extents und deren Länge sind ebenfalls in den File Headern gespeichert. Durch die Errechnung der Prüfsumme der ersten 256 Byte kann die Intigrität des Headers überprüft werden Convergent Technologies [1987].

3.4. Bad Block

Datenträger verfügen meistens ab Werk über defekte Sektoren. Um trotzdem eine korrekte Funktion zu gewährleisten, werden defekte Sektoren in einer Bad Block Datei gespeichert. Dadurch können fehlerhafte Sektoren während eines Speichervorgangs ausgelassen und fehlerfreie Sektoren zur Speicherung genutzt werden. Der Bad Block ist als die Systemdatei „BadBlk.Sys“ gespeichert und kann bis zu 128 defekte Sektoren verzeichnen Convergent Technologies [1987].

3.5. Allocation Bit Map

Jedes Bit der Allocation Bit Map repräsentiert die Verfügbarkeit eines Sektors auf einem Datenträger. Das Bit ist auf 1 gesetzt, wenn der Sektor verfügbar ist. Die Anfangsadresse der Allocation Bit Map wird im VHB verzeichnet und die Größe der Bit Map entspricht der Anzahl der Sektoren des Datenträgers. Es handelt sich hierbei um keine Datei. Convergent Technologies [1987]

3. BTOS/CTOS Dateisystem

4. Ansatz

Die digitale Forensik befasst sich vornehmlich mit der Analyse von aktuellen Systemen. Daher sind die verfügbaren Tools, welche die Arbeit erleichtern sollen, auf aktuelle Systeme und Formate abgestimmt. Trotzdem bleibt die grundlegende Herangehensweise die selbe. Die vorhandenen Informationen sind für die weitere Arbeit vollkommen ausreichend.

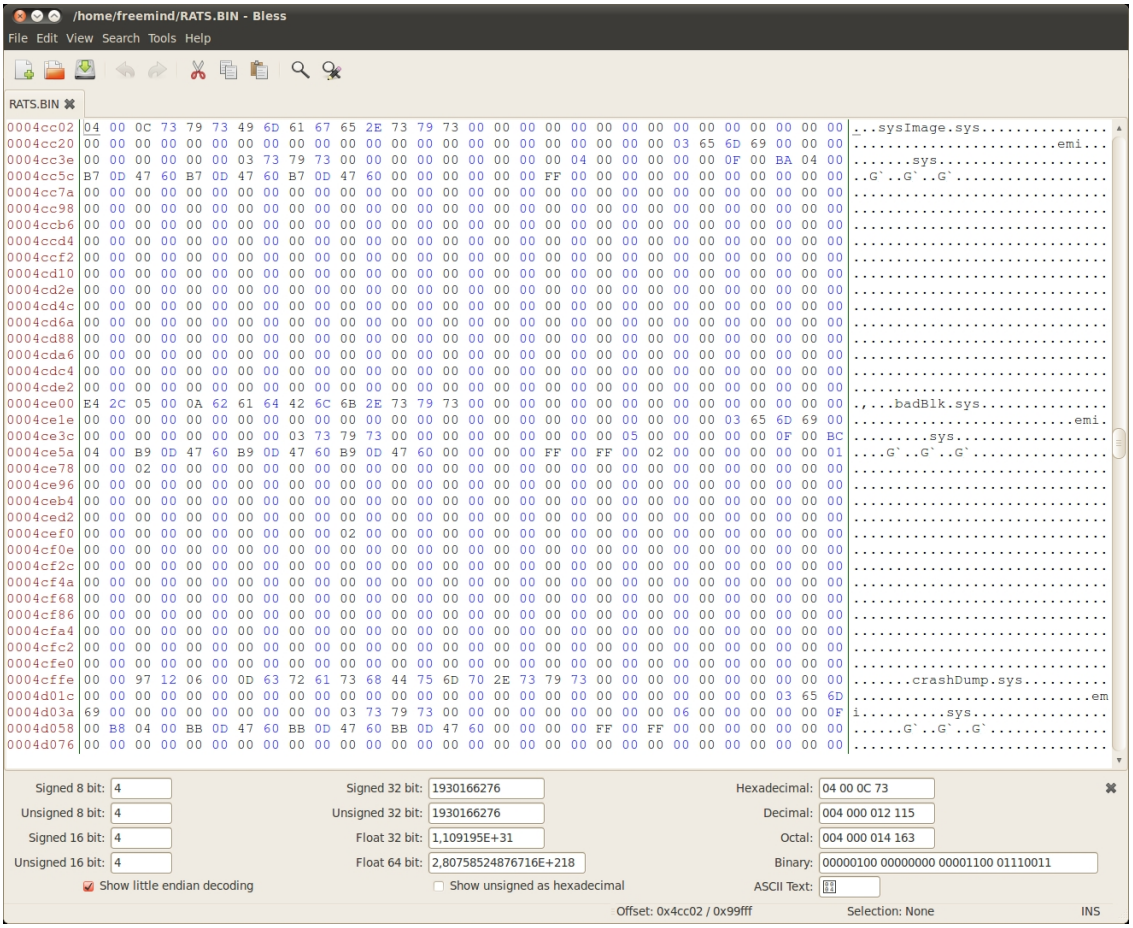


Abbildung 4.1.: Einträge der Systemdateien in den File Headern.

Als erstes wird eine Analyse des Dateisystems durchgeführt. Dabei wird überprüft, ob die Dateisystemspezifikation mit dem Dateisystem der Disketten übereinstimmt. Hierzu werden der VHB und die File Header mittels eines Hexeditors analysiert. In Abbildung **4.1** sind im FHB die für das BTOS/CTOS charakteristischen Betriebssystemdateien zu erkennen. Die Informationen aus den Dateisystemstrukturen liefern einen Weg zu den Metainformationen der Dateien. Die Metainformationen führen zu den einzelnen Disk Extents, welche die eigentliche Datei enthalten.

Es sind zwei Wege zu den Dateien möglich. Der erste Weg führt vom VHB zum MFD über die Verzeichnisinformationen zu den File Headern. Der Zweite führt vom VHB direkt in die File Header und somit zu den Dateien. Hier bietet sich der direkte Weg an. Bei diesem Vorgehen wird die Wahrscheinlichkeit minimiert, aufgrund des Alters der Datenträger aus beschädigten Bereichen zu lesen und damit das Ergebnis zu gefährden. Damit steht eine Strategie zum Vorgehen fest. Darauf basierend wird ein Programm erstellt, welches die Strukturen der Abbilder interpretiert, die Dateien in den Speicher lädt und in das Dateisystem des Hostcomputers integriert. Während dieses Vorgangs kommt es ausschließlich zu lesendem Zugriff auf die Abbilder. Es werden also keine Daten verändert. Die extrahierten Dateien können dann auf ihren Typ hin überprüft werden, um ein geeignetes Programm zu finden, welches den Inhalt richtig interpretiert. Sind die Dateisystemstrukturen zerstört, ist es letzendlich möglich, durch eine ASCII-Texterkennung beinhaltete Textfragmente zu retten.

5. Durchführung

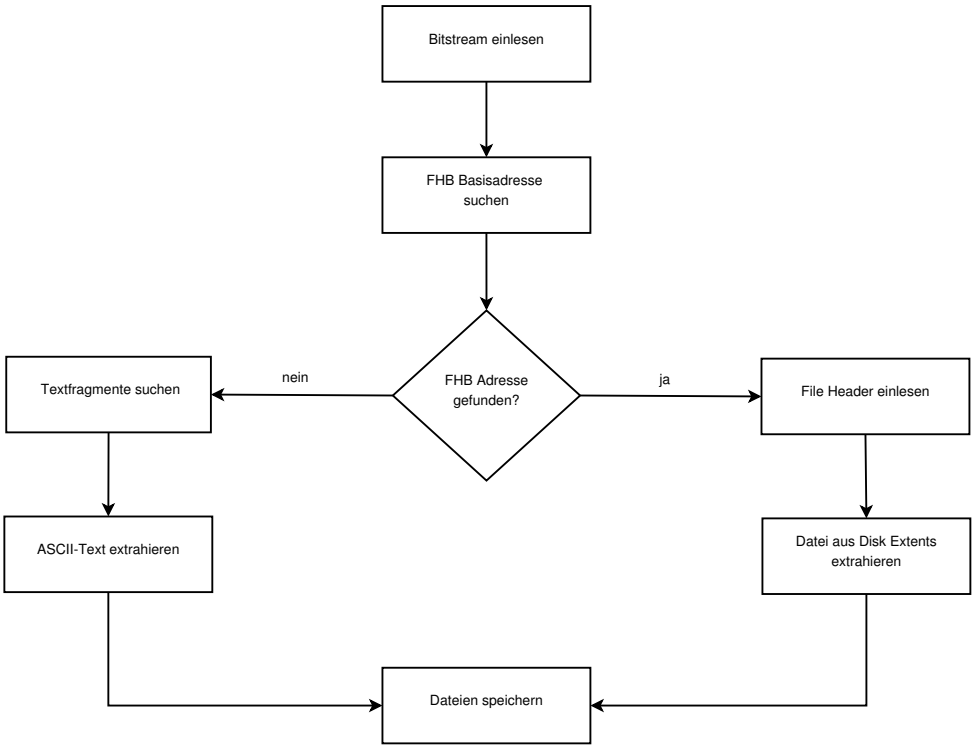


Abbildung 5.1.: Ablauf der Dateiextraktion.

Um einen Überblick zu erhalten, werden die Abbilder einer manuellen Analyse mit Hilfe eines Hexeditors unterzogen. Zunächst wird der VHB untersucht, der als Wurzelknoten für alle weiteren Strukturen dient. Darin sind die Zeiger auf das MFD und den FHB verzeichnet. Die ausgelesenen Zeiger führen bei allen Abbildern zu Adressen, welche größer sind als das Abbild selbst. Diese Erkenntnis legt nahe, dass es sich hierbei um einen Interpretationsfehler der Bytereihenfolge handeln muss. Die Zeiger liegen im Little Endian Format vor.

Neben den Metainformationen existieren in den File Headern für jede Datei zwei Arrays. Das erste Array enthält die Sektoradressen der zur Datei gehörigen Disk

Extents, und das zweite Array enthält die entsprechende Länge der Daten. Bei Textdateien ist ab dieser Adresse Text im Hexeditor erkennbar. Die Informationen aus der manuellen Analyse werden zur Erstellung eines Programms genutzt, dass die Dateiextraktion automatisch durchführt. Eine zeiteffiziente Lösung steht hier nicht im Vordergrund, da einzig das Auslesen der Dateien das Ziel ist und nicht die Erstellung eines Dateisystemtreibers.

Das Extraktionsprogramm wird in Python 2.6.5 geschrieben. Als Illustration des Ablaufs dient das Diagramm **5.1**. Das Programm kennt die beiden möglichen Basisadressen des VHBs. Aufgrund der Tatsache, dass es einen aktiven VHB und einen als Backup gibt, wird durch die Prüfsumme der intakte Block ausgewählt. Stimmt die Prüfsumme bei beiden nicht, so wird an den Positionen nach passenden Werten gesucht, an denen theoretisch die Adresse des FHBs zu finden ist. Als nächstes werden die File Header sequentiell bis zum Ende durchgegangen und enthaltene Dateien gespeichert. Dabei werden auch gelöschte Dateien, hier Dateien mit inaktivem Header, gesichert. Falls mehrere unterschiedliche, plausible File Header für eine Datei vorhanden sind, werden beide unter unterschiedlichen Namen abgespeichert. Nach der Speicherung werden Dateiname, Verzeichnis, Passwort, Änderungsdatum und Größe als Ausgabe angezeigt. Es werden nur reguläre Dateien gespeichert und keine Systemdateien.

Alter und Abnutzungserscheinungen der Disketten kann zu Abbildern führen, bei denen der Ansatz über den VHB und den FHB nicht mehr möglich ist. Dennoch können auch aus diesen Abbildern Daten zurückgewonnen werden. Da die richtige Interpretation von Zeichenkodierungen, die maximal 1 Byte je Zeichen verwenden, nicht von der Byte-Reihenfolge abhängig ist, können ASCII-kodierte Zeichenfolgen problemlos extrahiert werden. Diese Funktion wird angewendet, wenn die eigentlichen Dateisystemstrukturen keine verwertbaren Informationen zur Extraktion liefern.

Nach der Extraktion werden die erhaltenen Dateien mit dem Unix Befehl `file` auf deren Dateityp getestet. Das `file` Programm führt drei Tests in der folgenden Reihenfolge durch. Als erstes wird untersucht, ob es sich um eine Systemdatei handelt. Der zweite Test vergleicht den inneren Aufbau der Datei mit bekannten Strukturen regulärer Dateien. Zuletzt wird versucht, zwischen Binär- und Textdateien zu unterscheiden. Dieser Test kann einen Anhaltspunkt für den möglicherweise zugrunde liegenden Dateityp geben.

6. Ergebnisse

Es wurden von 99 Diskettenabbildern insgesamt 1889 Dateien aus den Jahren 1983 bis 1993 erfolgreich extrahiert. Davon wurden 1789 Dateien mit aktivem Header und 100 gelöschte Dateien wiederhergestellt. Die Typerkennung der Dateien durch `file` wird in Tabelle **6.1** aufgeführt.

Dateityp	Anzahl
data	1635
ASCII text	106
ASCII English text	15
ISO-8859 text	11
XPack DiskImage archive data	7
DBase 3 data file	5
FORTRAN program	2
Lisp/Scheme program text	2
Non-ISO extended-ASCII text	2
8086 relocatable (Microsoft)	1
ASCII C program text	1
Emacs v18 byte-compiled Lisp data	1
MS Windows icon resource	1

Abbildung 6.1.: Dateitypen der wiederhergestellten Dateien.

Eine allgemeine Beobachtung ist die unverschlüsselte Abspeicherung der Dateien, obwohl für Datenträger, Verzeichnisse und Dateien jeweils ein optionales Passwort gesetzt werden kann. Dieser Zugriffsschutz wird wohl in Form einer Authentifizierung vom Betriebssystem stattfinden.

Die größte Anzahl der Dateien sind Binärdateien und werden als „data“ erkannt. Dies ist die Ausgabe von `file`, wenn keiner der Tests erfolgreich anschlägt. Dieses Ergebnis ist nicht sehr überraschend, da es sich hierbei um relativ alte und nicht

sehr verbreitete Dateitypen handelt. Aufgrund von hohem Textanteil und wiederkehrenden Bytesequenzen ist bei 838 der Binärdateien anzunehmen, dass dies möglicherweise Dokumente eines Programms namens Word Processor sind. Dies war ein Textverarbeitungsprogramm, welches mit den BTOS/CTOS Betriebssystemen mitgelierfert wurde. Viele dieser Dateien sind Amtsakten, unter anderem mit Inhalten zu vergangenen Gerichtsprozessen. Weiterhin befinden sich unter den als „data“ erkannten Dateien 82 ausführbare Programme. Die Erkennung gelingt über die Dateiendung „.run“, welche bei BTOS/CTOS Betriebssystemen für eine ausführbare Datei steht Miller u. a. [1991]. Es werden weitere Dateien mit Textinhalt ermittelt, welche nicht in ein binäres Format eingebettet sind und somit von `file` auch erkannt werden. Davon sind 3 Dateien Programmcode der Sprachen Lisp und C. Außerdem werden 7 Dateien als XPack Archivdateien identifiziert. In 5 Fällen handelt es sich um Datenbankinhalte im DBase 3 Format. Eine genauere Analyse der Datenbankdateien erbrachte das Resultat, dass es sich hierbei um eine Fehlinterpretation von `file` handeln muss. 2 Dateien sind Fortran basierte Programme. Es wurde eine Datei mit Emacs Lisp Bytecode und ein Programm für die 8086 Plattform, dass wohl unter einem Microsoft DOS lauffähig ist, wiederhergestellt. Die Microsoft Icon Resource stellt sich bei genauer Betrachtung ebenfalls als Fehlerkennung heraus.

Die restlichen Binärdateien sind dem `file` Programm nicht bekannt oder sind teilweise zerstört, was eine erfolgreiche Erkennung unmöglich machen kann. Abgesehen von Dateien mit Textinhalt ist es bei den restlichen Dateien nicht so einfach möglich, den Inhalt zu interpretieren. Hierzu fehlen die zugehörigen Programme oder Dateispezifikationen. Als Beispiel einer Extraktion dient die Datei „joe“ aus dem Diskettensatz der US Küstenwache. In Abbildung **6.2** ist der Textinhalt bereits im Hexeditor sichtbar. Darauf folgt die Ausführung des Extraktionsprogramms mit der zugehörigen Abbilddatei als Input. Die Ausgabe wird von Abbildung **6.3** wieder gespiegelt. Unter Verwendung von Open Office 3.2 ist in Abbildung **6.4** ersichtlich, dass die Datei sogar teilweise richtig interpretiert wird.

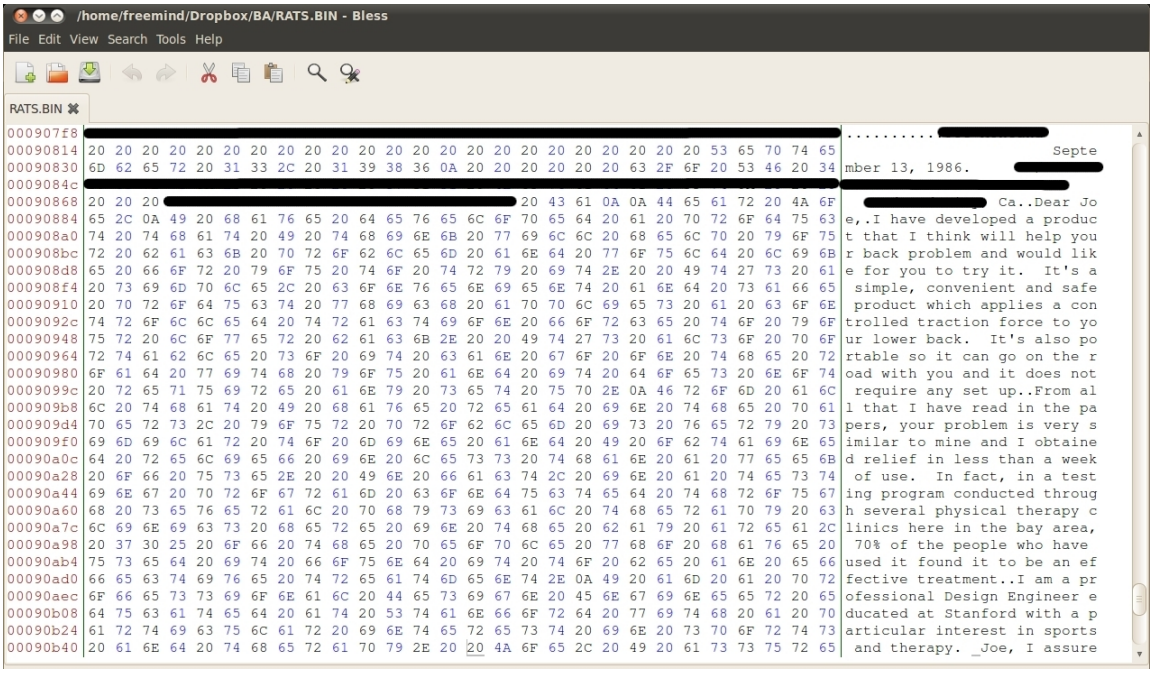


Abbildung 6.2.: Binärdatei im Hexeditor angezeigt.

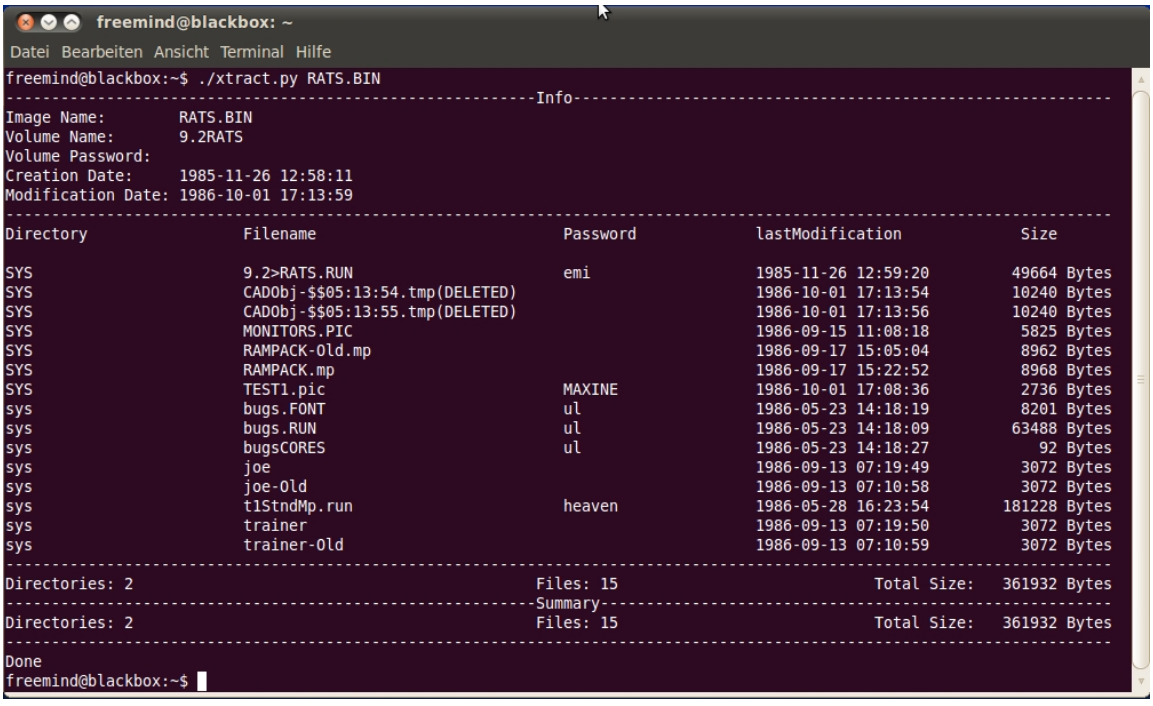


Abbildung 6.3.: Ausgabe des Extraktionsprogramms.

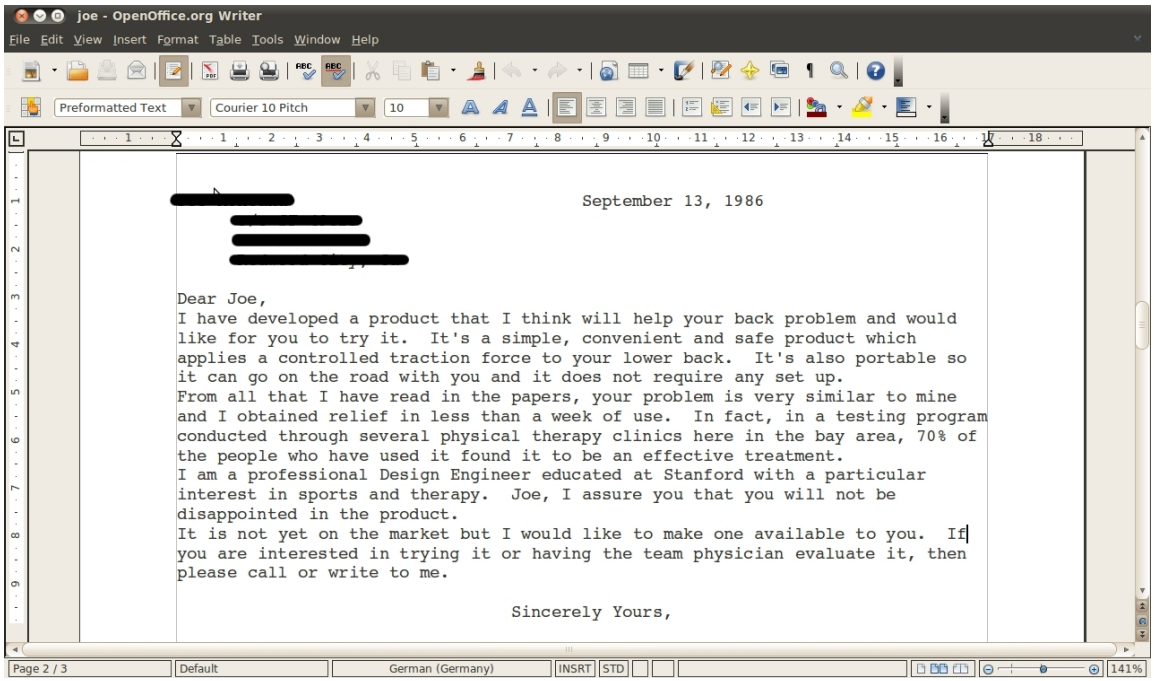


Abbildung 6.4.: Binärdatei interpretiert von Open Office 3.2.

7. Zusammenfassung und Ausblick

In dieser Arbeit wurde Anhand der Wiederherstellung von Dateien alter Datenträger gezeigt, wie eine Vorgehensweise bei Problemen dieser Art aussehen kann. Die Herausforderung ist in diesem Fall nicht nur die Rekonstruktion, sondern besonderst auch der Informationsmangel zu den alten Technologien. Können genügend Informationen zusammen getragen werden, ist es möglich, mit Methoden der digitalen Forensik Daten wieder zu gewinnen. Falls nicht genug Informationen vorhanden sind, kann es schwierig bis unmöglich werden, verwertbare Daten zu erhalten.

Wie sich in den Resultaten wiederspiegelt, ist die Wiederherstellung nicht der letzte Schritt zu verwertbaren Daten zu gelangen. Es müssen weitere Analysen durchgeführt werden, um den Dateityp zu erkennen und die Datei dementsprechend zu interpretieren. Dazu sind wiederum Informationen zu diesen Dateitypen notwendig, wenn der Dateiinhalt gerettet werden soll. Reine Textdateien sind relativ einfach zu erkennen und zu interpretieren, wobei Binärdateien je nach Alter und Verbreitung schwer identifizierbar sein können. Ohne ein geeignetes Programm, sind die Daten von nicht identifizierbaren Dateien für immer verloren.

Schaut man sich die rasante Entwicklung der Informationstechnologien an, so wird es zu einem schwerwiegenden Problem werden, die immer größer werdenden Datenmengen gegen das endgültige Vergessen zu schützen. Diesem Problem muss mit Techniken der digitalen Langzeitarchivierung begegnet werden, um das über Jahrtausende gesammelte Wissen, auf dem das gesamte Dasein der Menschheit beruht, für die Zukunft zu bewahren.

7. Zusammenfassung und Ausblick

Anhang

A. DVD

Dieser Arbeit ist eine DVD beigelegt. Sie enthält das Extraktionsprogramm „xtract.py“ und das Abbild einer 5.25 Zoll Diskette. Die Ordner auf der DVD müssen auf die Festplatte kopiert werden. Über die Konsole erfolgt der Aufruf des Programms im Programmordner durch: **„ ./xtract.py <image file> “**

Literaturverzeichnis

Bartsch 2011
Bartsch, Christian: *Kryoflux - High Definition Flux Sampler for USB*. http://info-coach.fr/atari/hardware/devices/kryoflux/kryoflux_stream_protocol.pdf. 2011. – [Online; letzter Aufruf 21.05.2012]

Carrier 2005
Carrier, Brian: *File System Forensic Analysis*. 1. Addison Wesley Professional, 2005. – ISBN 978-0321268174

Convergent Technologies 1987
Convergent Technologies: *FILESYS*. http://www.oocities.org/siliconvalley/pines/4011/misc-doc/FILESYS.htm. 1987. – [Online; letzter Aufruf 10.04.2012]

Frank 2005
Frank, Andreas: *5.25"Diskette without sleeve*. http://de.wikipedia.org/w/index.php?title=Datei:5.25%22-Diskette.jpg&filetimestamp=20080327183132. 2005. – [Online; letzter Aufruf 21.05.2012]

Miller u. a. 1991
Miller, Edna I. ; Croock, Jim ; Loy, June: *Exploring CTOS*. Prentice Hall, 1991. – ISBN 0-13-297342-1

Sietmann 2006
Sietmann, Richard: *Kulturelles Erbe in Gefahr*. http://www.heise.de/newsticker/meldung/Kulturelles-Erbe-in-Gefahr-124097.html. 2006. – [Online; letzter Aufruf 21.05.2012]

Tanenbaum 2001
Tanenbaum, Andrew S.: *Modern Operating Systems*. 2. Prentice Hall PTR, 2001. – ISBN 0-13-031358-0

Tanenbaum und Goodman 1987
TANENBAUM, Andrew S. ; GOODMAN, James: *Computerarchitektur*. 4. Prentice Hall, München, 1987. – ISBN 978-3827295736